

# MICROSERVICES DESIGN

# Outline

- ▶ Microservices Design Patterns Overview
- ▶ Microservice Decomposition Patterns
- ▶ Domain Driven Design (DDD) Overview

# MIRCOSERVICES PATTERNS

# Microservices Patterns

- <http://microservices.io/patterns>

# DOMAIN DRIVEN DESIGN

# What is Domain Driven Design (DDD)

- Aims to ease the creation of complex applications
- Connects the related pieces of the software into an ever-evolving model.
- DDD focuses on three core principles:



Focus on the core domain and domain logic.

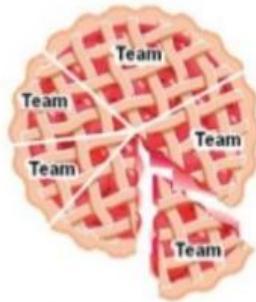
Find complex designs on models of the domain.

Constantly collaborate with domain experts to improve the application model and to resolve any domain-related issues.

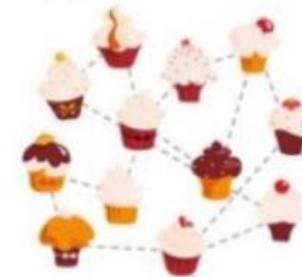
# Why Domain Driven Design (DDD)

With the evolution of microservices, the traditional design SOA concept of identifying the services and expose based on reuse was not sufficient since microservices is a business driven concept

There exists the need for a design methodology which brings business and services together. This is where the domain driven design concept fits well and helps in designing Microservices.

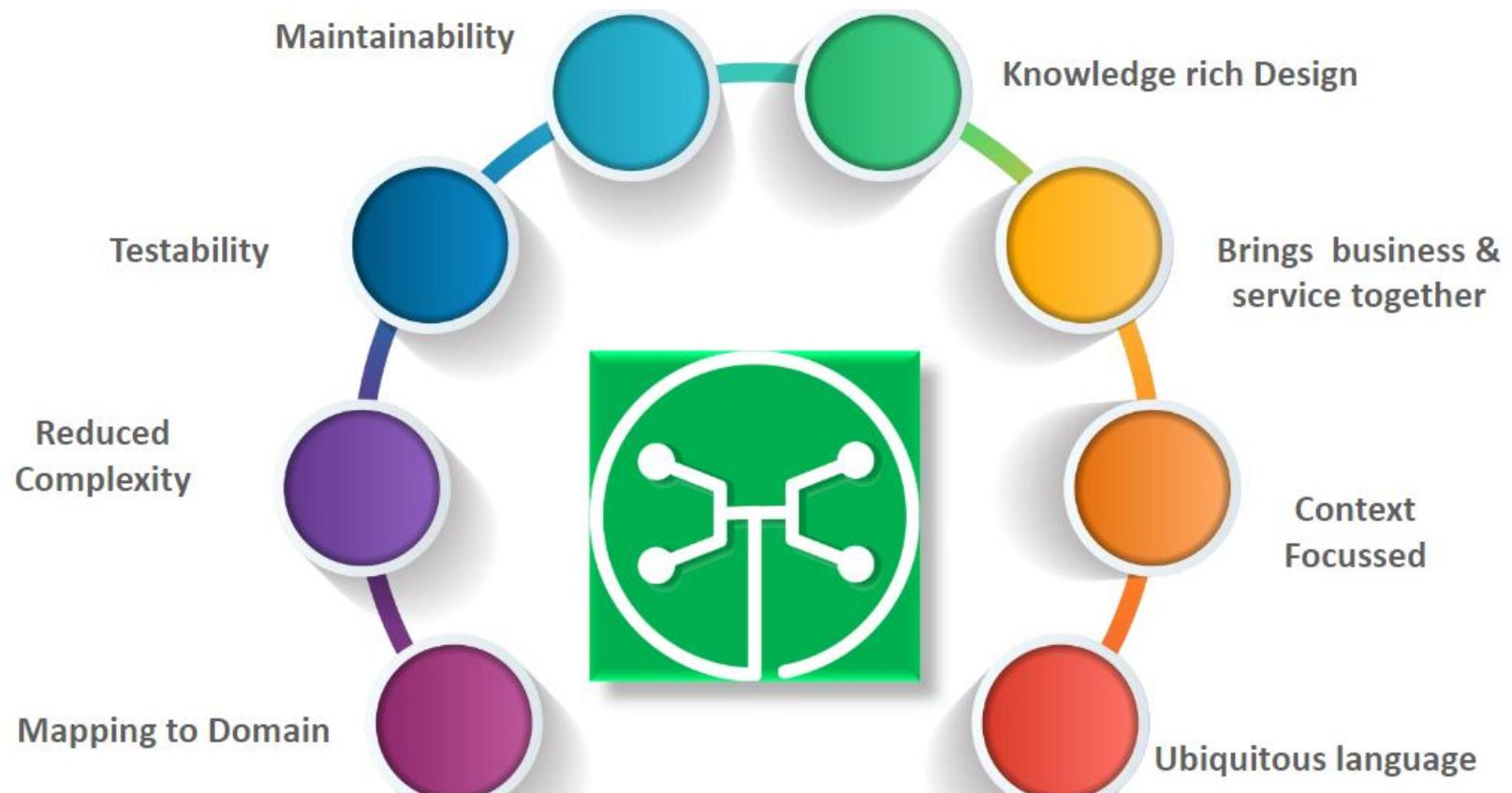


Traditional SOA



Microservices

# Why Domain Driven Design (DDD)



# What is a Domain

Domain



Sphere of knowledge and activity around which the application logic revolves

Domain logic/  
Business logic



Higher-level rules for how business objects interact with one another to create and modify modeled data.

# Domain Driven Concepts

## Context

The setting in which a word or statement appears that determines its meaning.

Statements about a model can only be understood in a context.

## Ubiquitous Language

A language structured around the domain model and used by all team members to connect all the activities of the team with the software.



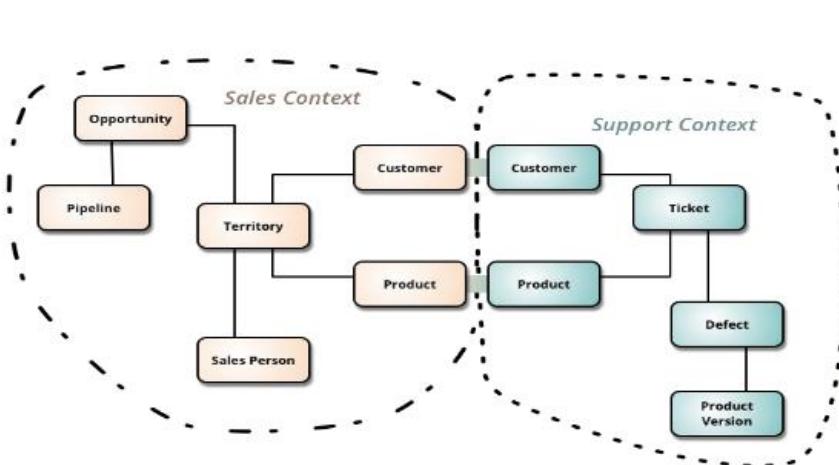
## Model

A system of abstractions that describes selected aspects of a domain and can be used to solve problems related to that domain.

## Bounded Context

A description of a boundary (typically a subsystem, or the work of a specific team) within which a particular model is defined and applicable.

# What is Bounded Context



Large models divided into different Bounded Contexts



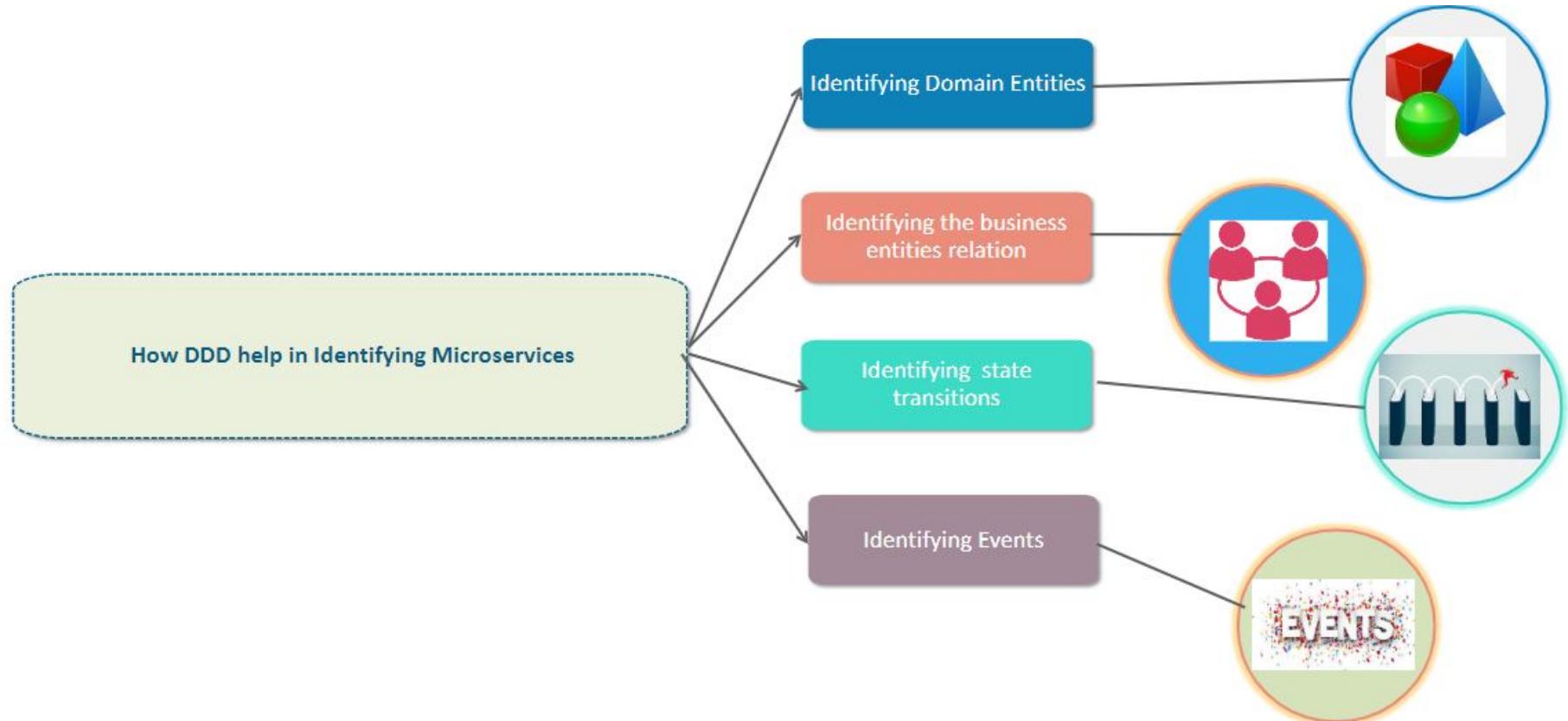
For large systems, having one unified model becomes very complex, which is why it is broken in smaller bonded context, e.g. sales and support domains in the left picture.

Bounded Context is a central pattern in Domain-Driven Design



Explicit about their inter-relationships

# How DDD help in Identifying Microservices



# Thank You!